Анализ транспортной сети МЦД

Выполнили: Сычева Ирина Артемовна и Синаков Арсений Александрович

Почему МЦД?

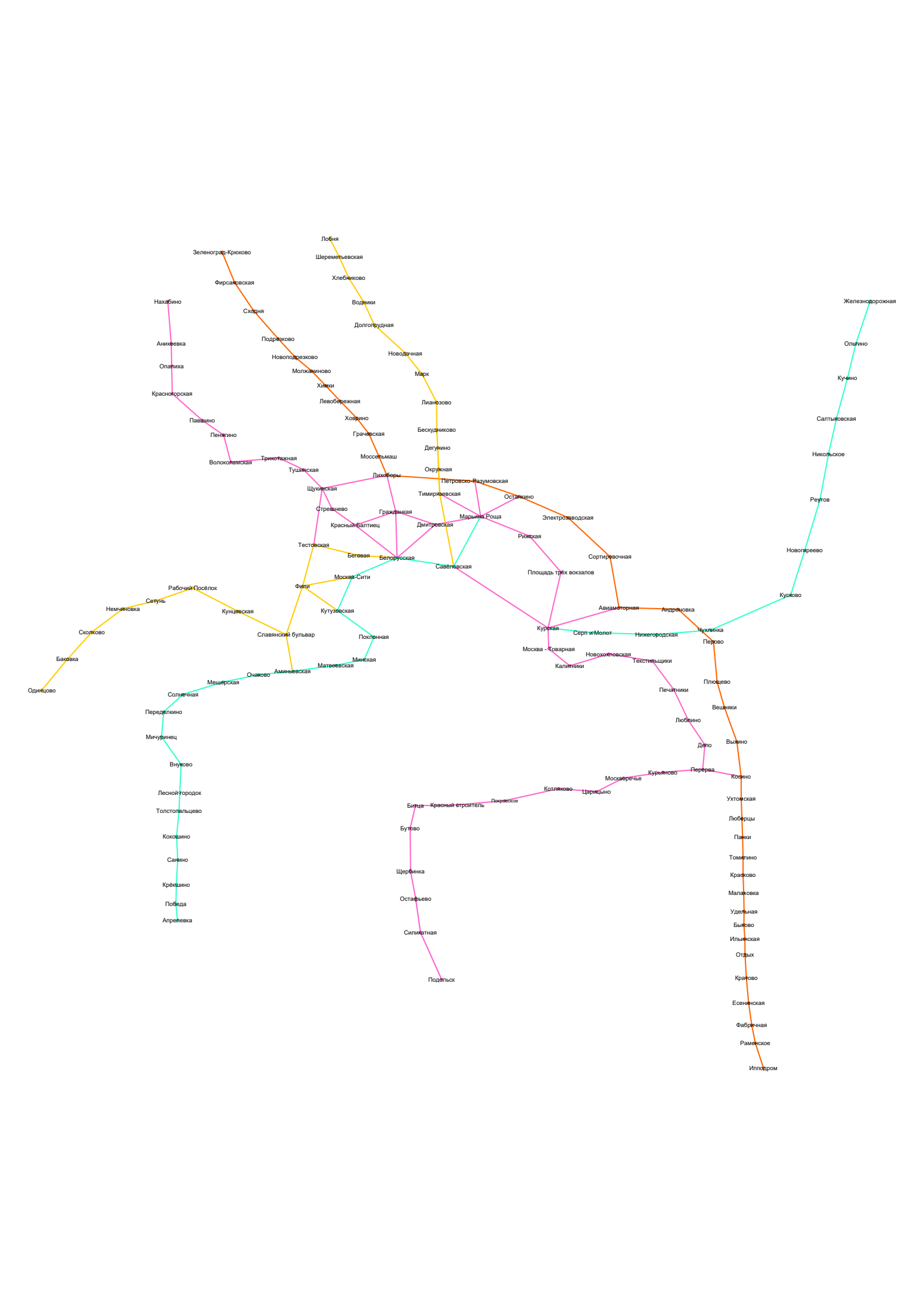
Транспортная система крупного мегаполиса представляет собой сложную сетевую структуру. И эффективность ее функционирования напрямую зависит от оптимальности связей между узлами. Поэтому такой анализ может помочь решить несколько проблем, с которыми сталкиваются развивающиеся сети. Например, оптимизировать пассажиропоток, проверить устойчивость инфраструктуры.

Источники и построение

При построении графа МЦД, мы пользовались оф. сайтом города Москвы. Но также после построения, мы решили немного видоизменить граф и добавить в него новые ребра для выполнения 6-8го задания, так как степени вершин были слишком маленькие. А сама сеть строилась по принципу "станция - станция". Ребро между двумя вершинами означает, что есть транспортное сообщение какое-то между станциями. Изначально мы также граф сделали невзвешенным и неориентированным. Но после мы добавили рандомные веса от 1 до 5 для выполнения задания. Веса тут означают "затраты на время".

Сам граф

Можете взглянуть ниже и посмотреть, какая красота у нас получилась :)


Лобня
Зеленоград-Крюково
Шереметьевская
Фирсановская
Хлебниково
Нахабино
Водники
Железнодорожная
Сходня
Долгопрудная
Ольгино
Подрезково
Аникеевка
Новодачная
Новоподрезково
Кучино
Опалиха
Молжаninово
Марк
Химки
Красногорская
Левобережная
Лианозово
Салтыковская
Павшино
Ховрино
Бескудниково
Пенягино
Грачевская
Дегунино
Никольское
Волоколамская
Трикотажная
Моссельмаш
Тушинская
Окружная
Лихоборы
Петровско-Разумовская
Щукинская
Тимирязевская
Останкино
Реутов
Стрешнево
Гражданская
Марьина Роща
Электрозаводская
Красный Балтиец
Дмитровская
Рижская
Тестовская
Беговая
Белорусская
Сортировочная
Новогиреево
Савеловская
Москва-Сити
Площадь трёх вокзалов
Рабочий Посёлок
Филя
Сетунь
Кусково
Немчиновка
Кутузовская
Авиамоторная
Андрюшовка
Кунцевская
Курская
Чухлинка
Сколково
Серп и Молот
Нижегородская
Славянский бульвар
Поклонная
Перово
Баковка
Москва - Товарная
Новохохловская
Минская
Матвеевская
Калитники
Текстильщики
Аминьевская
Очаково
Плющево
Мещерская
Печатники
Одинцово
Солнечная
Вешняки
Переделкино
Люблино
Мичуринец
Депо
Выхино
Внуково
Курьяново
Перерва
Косино
Москворечье
Лесной городок
Котляково
Царицыно
Ухтомская
Толстопальцево
Покровское
Битца
Красный строитель
Люберцы
Бутово
Пятки
Кокошкино
Томилино
Щербинка
Санино
Красково
Крёкшино
Остафьево
Малаховка
Победа
Удельная
Силикатная
Апрелевка
Быково
Ильинская
Подольск
Отдых
Кратово
Есенинская
Фабричная
Раменское
Ипподром

После этого мы получили статистические характеристики сети через gephi.

Мы узнали плотность графа, среднюю степень вершины, диаметр сети и прочее.

Ниже будут списки топ 5 узлов по каждой метрике:

Степень

1. Марьина Роща 6.0
2. Белорусская 6.0
3. Курская 5.0
4. Щукинская 4.0
5. Тимирязевская 4.0

RageRank

1. Марьина Роща 0.01648
2. Белорусская 0.01472
3. Курская 0.01216
4. Баковка 0.01214
5. Фили 0.01197

Betweenness Centrality

1. Курская 0.5197
2. Савёловская 0.5004
3. Белорусская 0.3788
4. Авиамоторная 0.3097
5. Андроновка 0.2931

Closeness Centrality

1. Савёловская 0.1169
2. Курская 0.1151
3. Белорусская 0.1151
4. Марьина Роща 0.1109
5. Авиамоторная 0.1097

Harmonic Closeness Centrality

1. Белорусская 0.2078
2. Савёловская 0.2063
3. Марьина Роща 0.2006
4. Курская 0.1989
5. Гражданкая 0.1867

Eccentricity

1. Андроновка 21.0
2. Авиамоторная 21.0
3. Калитники 22.0
4. Новохохловская 22.0
5. Перово 22.0

Сделаем какие-то выводы:

Марьина Роща и Белорусская имеют самую высокую центральность ну и степень, что говорит о том, что они работают как крупные пересадочные хабы.

Курская, на пример, показывает высочайшее посредничество, но сама обладает степенью 5, то есть она выступает как мост между сегментами сети

Савеловская обладает высокую близость и посредничество, поэтому она позиционирует себя как центр сети

Больше 80% станций имеют степень 2 и ниже. Это говорит об огромной зависимости от ключевых узлов

Выводы:

Курская нуждается в альтернативных маршрутах для снижения нагрузки с крит. моста. Стоит создать дублирующие связи в обход этой станции. Если с ней что-то станет, то будет плохо всей сети.

Ниже будут графики с метриками и также раскрашенный по определенному образу граф в зависимости от метрики.

PageRank Report

Parameters:

Epsilon = 0.001
Probability = 0.85

Results:

Algorithm:

Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999) *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab.

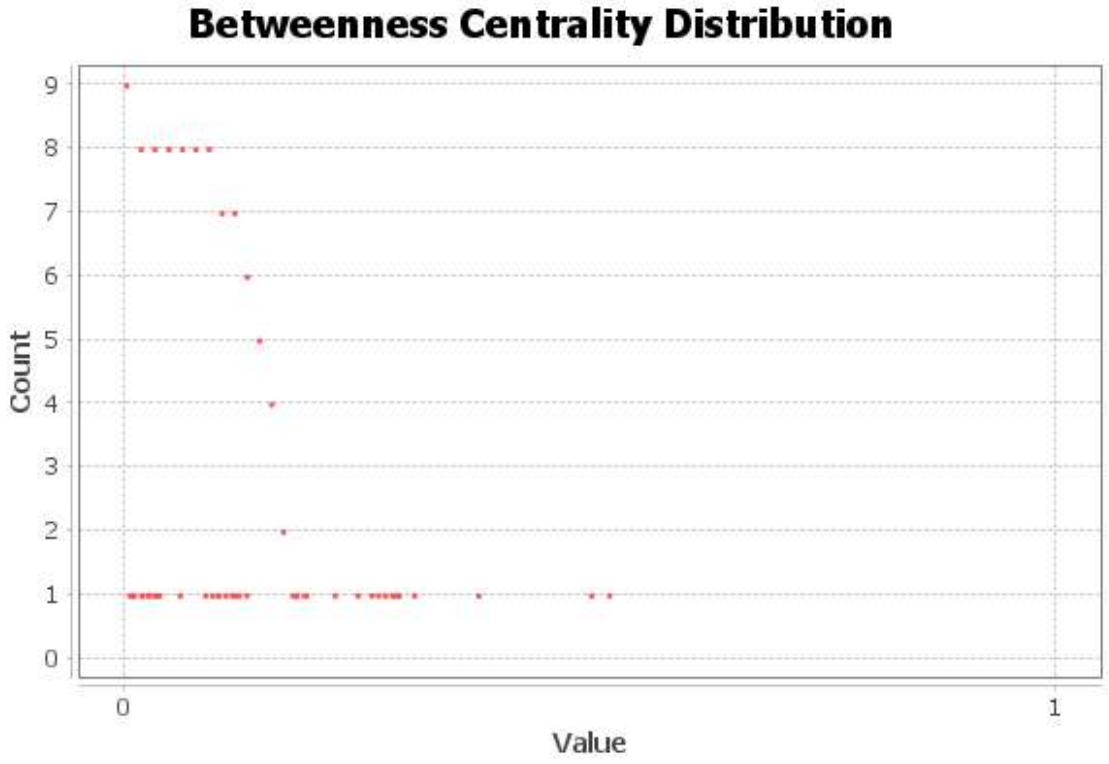
Graph Distance Report

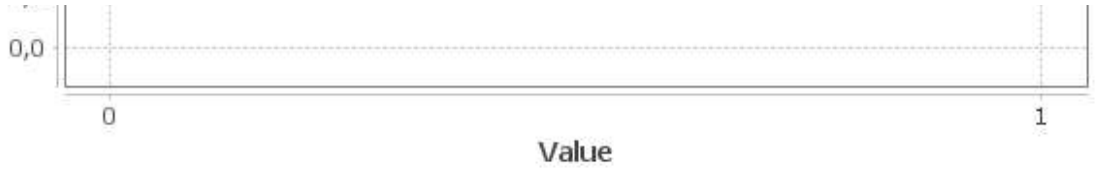
Parameters:

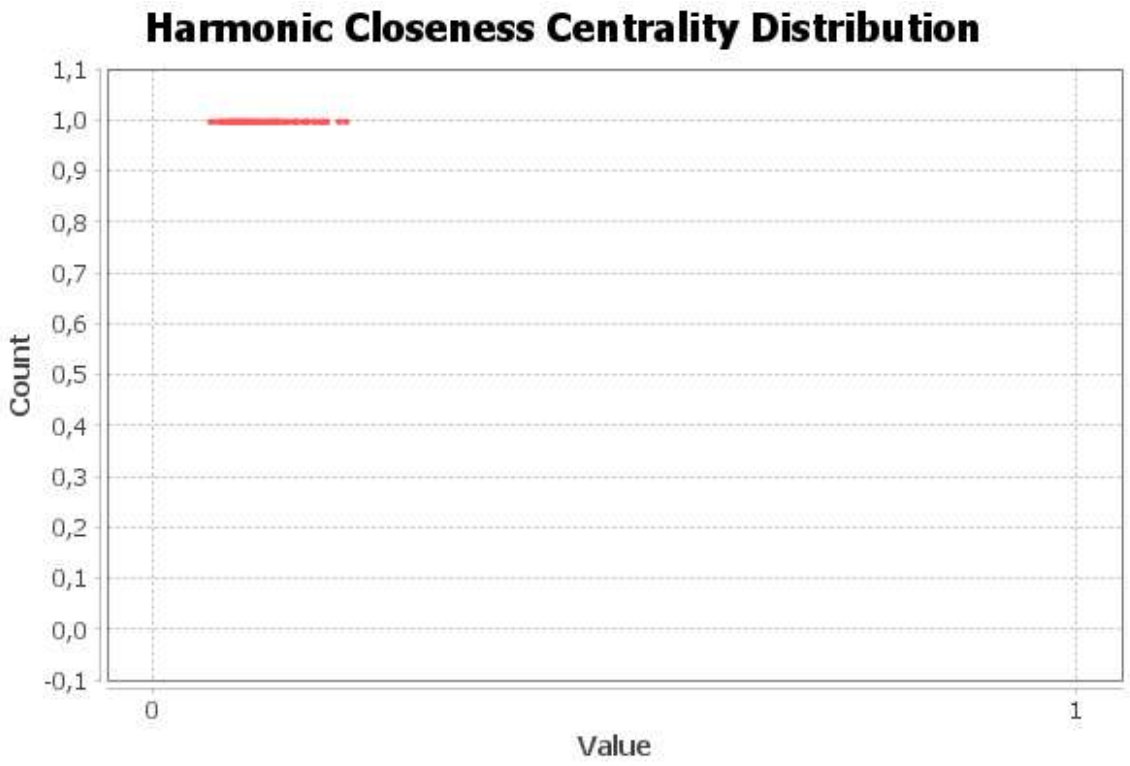
Network Interpretation: undirected

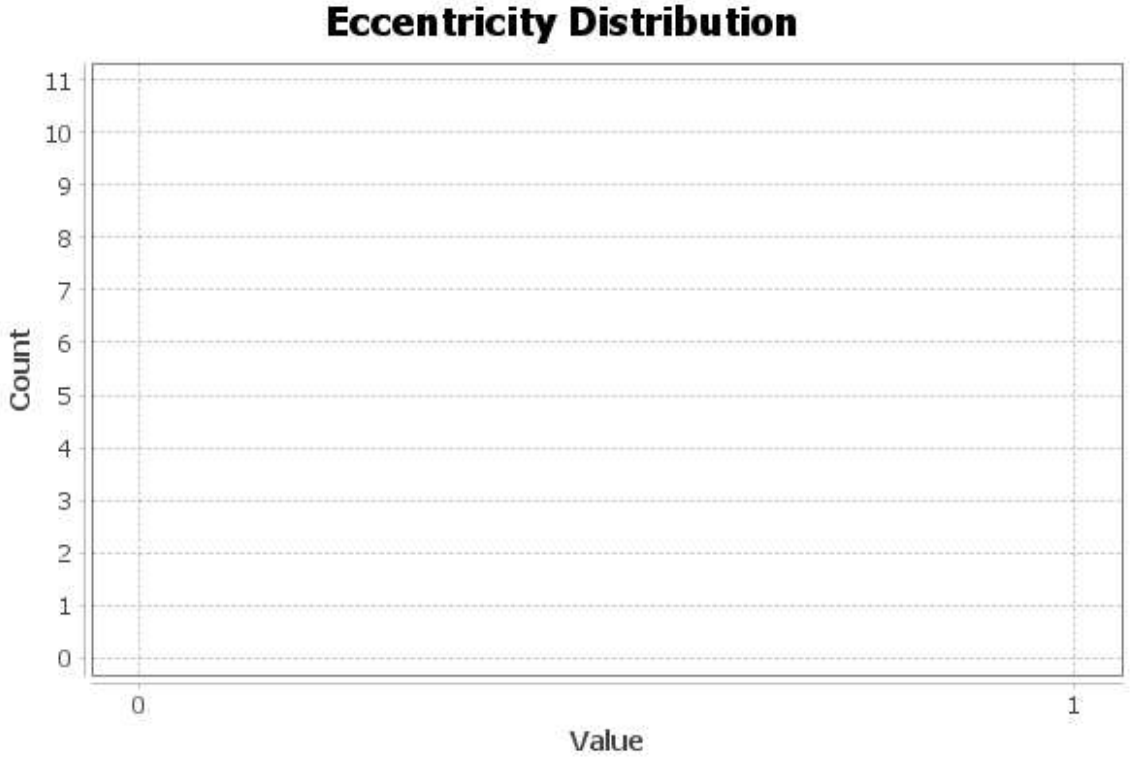
Results:

Diameter: 41
Radius: 21
Average Path length: 14.059424544920729




0,0
0
1
Value


Harmonic Closeness Centrality Distribution
1,1
1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
-0,1
Count
0
1
Value

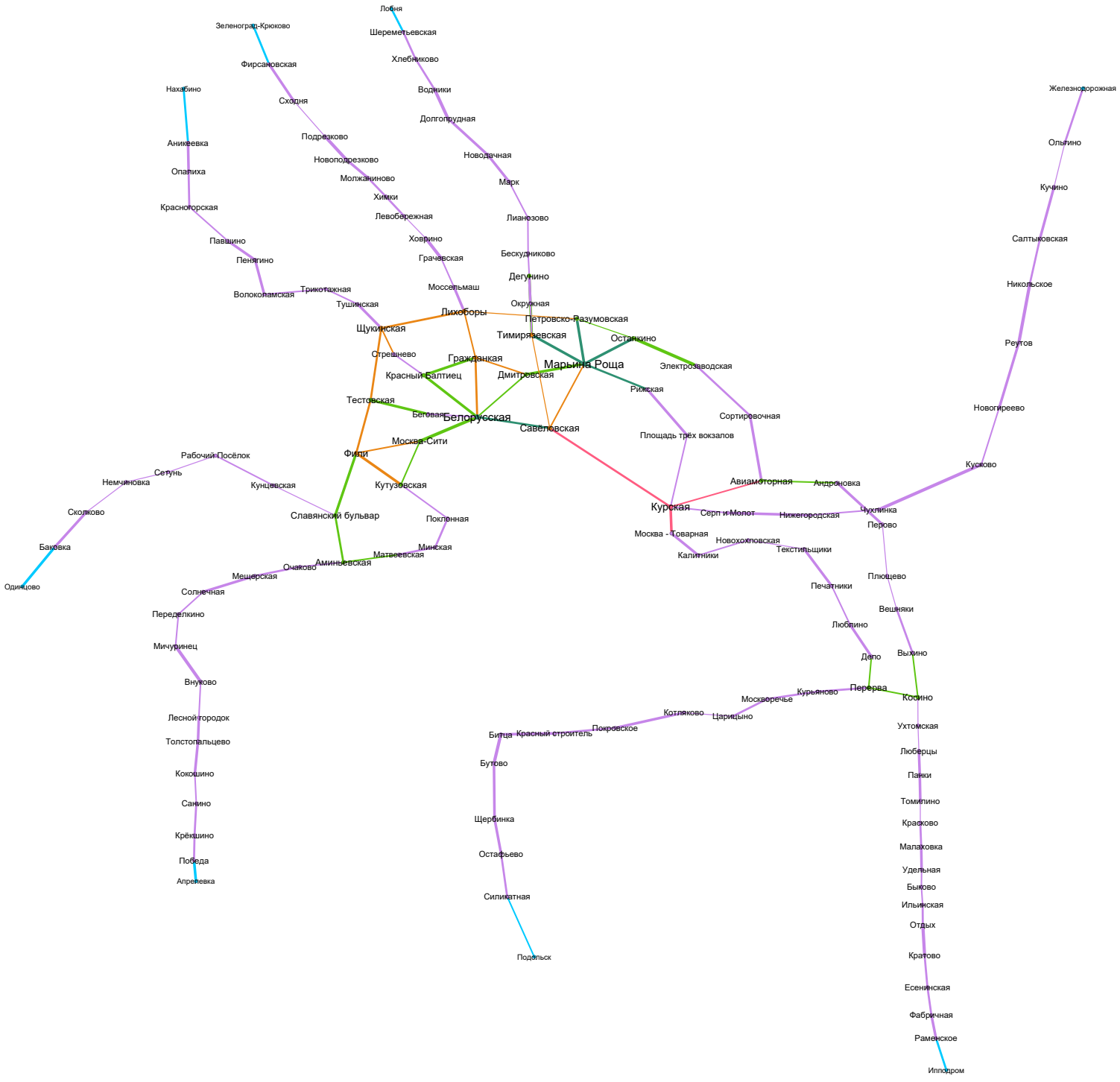

Eccentricity Distribution
11
10
9
8
7
6
5
4
3
2
1
0
Count
0
1
Value

Algorithm:

Ulrik Brandes, *A Faster Algorithm for Betweenness Centrality*, in Journal of Mathematical Sociology 25(2):163-177, (2001)

Суммарная мощность

Degree



Добавили веса на ребра

Closeness Centrality

Betweenness Centrality

RageRank


Любня
Зеленоград-Крюково
Шереметьевская
Фирсановская
Хлебниково
Нахабино
Водники
Железнодорожная
Сходня
Долгопрудная
Ольгино
Подрезково
Аникеевка
Новодачная
Новоподрезково
Кучино
Опалиха
Марк
Молжаninово
Химки
Красногорская
Левобережная
Лианозово
Салтыковская
Павшино
Ховрино
Грачевская
Бескудниково
Пенягино
Дегунино
Никольское
Трикотажная
Мосэльмаш
Волоколамская
Тушинская
Окружная
Лихоборы
Петровско-Разумовская
Щукинская
Тимирязевская
Останкино
Ревтов
Стрешнево
Гражданская
Марьина Роща
Электрозаводская
Красный Балтиец
Дмитровская
Римская
Тестовская
Новопеределкино
Беговая
Белорусская
Сортировочная
Савеловская
Площадь трёх вокзалов
Москва-Сити
Рабочий Посёлок
Фили
Кунцево
Сетунь
Немчиновка
Кутузовская
Авиамоторная
Андроновка
Сколково
Славянский бульвар
Курская
Серп и Молот
Чухлинка
Нижегородская
Перово
Баковка
Поклонная
Москва - Товарная
Новохохловская
Минская
Калитники
Текстильщики
Матвеевская
Аминьевская
Очаково
Плющево
Мещерская
Печатники
Одинцово
Солнечная
Вешняки
Переделкино
Люблино
Мичуринец
Депо
Выхино
Внуково
Перерва
Курьяново
Косино
Москворечье
Котляково
Царицыно
Лесной городок
Покровское
Ухтомская
Красный строитель
Толстопальцево
Бутово
Люберцы
Кокошкино
Панки
Щербинка
Томилино
Сакино
Красково
Остафьево
Малаховка
Крёкшино
Удельная
Силикатная
Победа
Быково
Ильинская
Апрелевка
Отдых
Подольск
Кратово
Есенинская
Фабричная
Раменское
Ипподром

Для выполнения 6-го задания, мы выбрали для анализа 3 вершины:

- Белорусская
- Марьина роща
- Курская

У Белорусской и Марьиной рощи степень вершины 6, у Курской же 5.

После этого мы выписали вершины, с которыми они соединены.

Например, для Белорусской:

Соседи и веса:

- Беговая $\rightarrow$ 2
- Москва-сити $\rightarrow$ 5
- Савеловская $\rightarrow$ 3
- Дмитровская $\rightarrow$ 2
- Гражданская $\rightarrow$ 3
- Красный балтиец $\rightarrow$ 4

Сумма всех весов: $2 + 5 + 3 + 2 + 3 + 4 = 19$. Тогда квота 30% от $19 = 5.7$.

Для Марьиной рощи:

Соседи и веса:

- Тимирязевская $\rightarrow$ 4
- Останкино $\rightarrow$ 4
- Петровско-разумовская $\rightarrow$ 4
- Дмитровская $\rightarrow$ 4
- Савеловская $\rightarrow$ 2
- Рижская $\rightarrow$ 3

Аналогично рассчитаем квоту и получим: $21 \cdot 0.3 = 6.3$

Для Курской:

Соседи и веса:

- Савеловская $\rightarrow$ 3
- Площадь трех вокзалов $\rightarrow$ 2
- Москва-товарная $\rightarrow$ 4
- Авиамоторная $\rightarrow$ 2
- Серп и молот $\rightarrow$ 2

Тут также рассчитаем квоту: $13 \cdot 0.3 = 3.9$

Для рассчета индексов bundle и pivotal мы решили написать простенький код на питоне. Его суть проста:

- находим крит. группы
- определяем ключевых игроков
- рассчитываем индексы

Крит. группы ищем через перебор всех возможных комбинаций соседних станций. Группа считается крит., если сумма весов $>$ квоты! Для каждой такой крит. группы проверяем, останется ли она крит. без каждой станции. Если нет, то станция считается ключевой для этой группы. Ну а после считаем индексы:

- Bundle = $\frac{\text{число групп, где станция ключевая}}{\text{всего крит. групп}}$
- Pivotal = $\max\left(\frac{\text{вес станции}}{\text{суммавесовгруппы}}\right)$

Ну и если bundle высокий, то это значит, что станция незаменима. Высокий pivotal значит доминацию станции в группах!

Вот такой код у нас получился:

```
import itertools


def calc_bundle_pivot(neighbors: dict, quota_val: float):
    stations = list(neighbors.keys())
    n = len(stations)
    crit_groups = []

    for size in range(1, n + 1):
        for group in itertools.combinations(stations, size):
            total_w = sum(neighbors[station] for station in group)
            if total_w > quota_val:
                crit_groups.append((group, total_w))

    key_count = {station: 0 for station in stations}
    pivot_vals = {station: [] for station in stations}

    for group, total_w in crit_groups:
        group_list = list(group)
        for station in group_list:
            new_w = total_w - neighbors[station]
            if new_w <= quota_val:
                key_count[station] += 1
                contrib = neighbors[station] / total_w
                pivot_vals[station].append(contrib)

    total_crit = len(crit_groups)
    bundle_idx = {station: count / total_crit for station, count in
    pivot_idx = {station: max(contribs) if contribs else 0 for stat

    return bundle_idx, pivot_idx, total_crit


bel_data = {
    'Беговая': 2,
    'Москва-сити': 5,
    'Савеловская': 3,
    'Дмитровская': 2,
    'Гражданская': 3,
    'Красный балтиец': 4
}
bel_quota = 5.7

mar_data = {
    'Тимирязевская': 4,
    'Останкино': 4,
    'Петровско-Разумовская': 4,
    'Савеловская': 2,
    'Дмитровская': 4,
    'Рижская': 3
}
mar_quota = 6.3
```

```
kur_data = {
    'Савеловская': 3,
    'Площадь трех вокзалов': 2,
    'Москва-товарная': 4,
    'Авиамоторная': 2,
    'Серп и молот': 2
}
kur_quota = 3.9

b_bel, p_bel, total_bel = calc_bundle_pivot(bel_data, bel_quota)
print("БЕЛОРУССКАЯ:")
for station in bel_data:
    print(f"   {station:30} Bundle: {b_bel[station]:.3f}  Pivotal: 

b_mar, p_mar, total_mar = calc_bundle_pivot(mar_data, mar_quota)
print("\nМАРЬИНА РОЩА:")
for station in mar_data:
    print(f"   {station:30} Bundle: {b_mar[station]:.3f}  Pivotal: 

b_kur, p_kur, total_kur = calc_bundle_pivot(kur_data, kur_quota)
print("\nКУРСКАЯ:")
for station in kur_data:
    print(f"   {station:30} Bundle: {b_kur[station]:.3f}  Pivotal: 

print("\nНаиболее важные станции:")
print(f"   БЕЛОРУССКАЯ:", max(bel_data, key=lambda x: b_bel[x]))
print(f"   МАРЬИНА РОЩА:", max(mar_data, key=lambda x: b_mar[x]))
print(f"   КУРСКАЯ:", max(kur_data, key=lambda x: b_kur[x]))
```

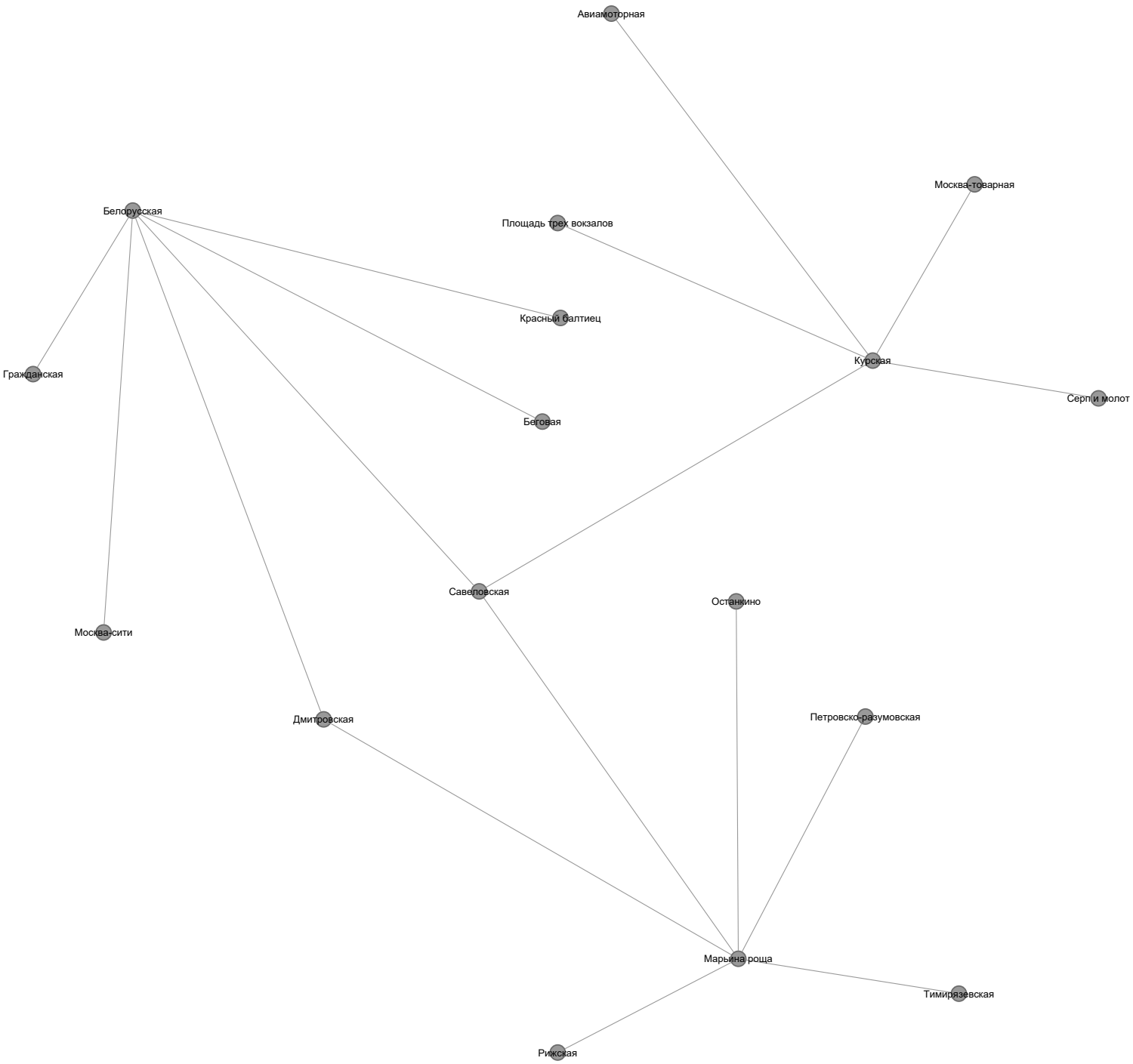
Перейдем к тому, что он вывел и какие индексы мы получили:

```
БЕЛОРУССКАЯ:
   Беговая                      Bundle: 0.077  Pivotal: 0.333
   Москва-сити                  Bundle: 0.192  Pivotal: 0.714
   Савеловская                  Bundle: 0.115  Pivotal: 0.500
   Дмитровская                  Bundle: 0.077  Pivotal: 0.333
   Гражданская                  Bundle: 0.115  Pivotal: 0.500
   Красный балтиец              Bundle: 0.192  Pivotal: 0.667

МАРЬИНА РОЩА:
   Тимирязевская                Bundle: 0.154  Pivotal: 0.571
   Останкино                    Bundle: 0.154  Pivotal: 0.571
   Петровско-Разумовская        Bundle: 0.154  Pivotal: 0.571
   Савеловская                  Bundle: 0.000  Pivotal: 0.000
   Дмитровская                  Bundle: 0.154  Pivotal: 0.571
   Рижская                      Bundle: 0.154  Pivotal: 0.429

КУРСКАЯ:
   Савеловская                  Bundle: 0.111  Pivotal: 0.600
   Площадь трех вокзалов        Bundle: 0.111  Pivotal: 0.500
   Москва-товарная              Bundle: 0.185  Pivotal: 1.000
   Авиамоторная                 Bundle: 0.111  Pivotal: 0.500
   Серп и молот                 Bundle: 0.111  Pivotal: 0.500

Наиболее важные станции:
   БЕЛОРУССКАЯ: Москва-сити
   МАРЬИНА РОЩА: Тимирязевская
   КУРСКАЯ: Москва-товарная
```

Ну и как мы видим, Москва-товарная, например, самая критическая станция, ибо pivotal = 1. Москва-сити и Красный балтиец - ключевые для Белорусской

Ну и отсюда можно сделать некие выводы и дать рекомендации. Например, при планировании ремонтов на Москва-сити может потребоваться разработка сложных схем обхода пассажиропотоков. А вот работы на Беговой и Дмитровской могут проводиться с минимальным влиянием на сеть. Касаемо Марьиной рощи, там есть равномерное распределение влияния между 4 станции, то есть эта сеть обладает высокой устойчивостью.

Также в gephi мы нарисовали данный граф.


Авиамоторная
Москва-товарная
Белорусская
Площадь трех вокзалов
Красный балтиец
Курская
Гражданская
Серп и молот
Беговая
Савеловская
Останкино
Москва-сити
Дмитровская
Петровско-разумовская
Марьина роща
Тимирязевская
Рижская

Задание 7

Мы разбили в gephi нашу сеть на сообщества. Через $statics \rightarrow modularity$. После я экспортировал $.csv$ файл с $nodes$ для анализа и нахождения самого большого сообщества!

```
import pandas as pd

df = pd.read_csv('modularity.csv', encoding='cp1251', delimiter=';')

community_counts = df['Modularity class'].value_counts()
print(f"Распределение: \n")
print(community_counts)
```

Тут мы получаем наше распределение по сообществам, откуда мы узнаем, что во 2-ом сообществе 16 вершин, в 5-ом 13 и т.д.

Полный вывод:

```
Распределение:

Modularity class
2     16
5     13
1     12
9     12
6     11
0      9
3      9
4      8
10     8
8      7
7      7
12     7
11     5
13     5
```

Прекрасно, мы поняли, что самое большое сообщество - это второе:

```
largest_community = community_counts.index[0]
largest_size = community_counts.iloc[0]

print(f"класс: {largest_community}, размер: {largest_size}")
```

Вывод:

```
класс: 2, размер: 16
```

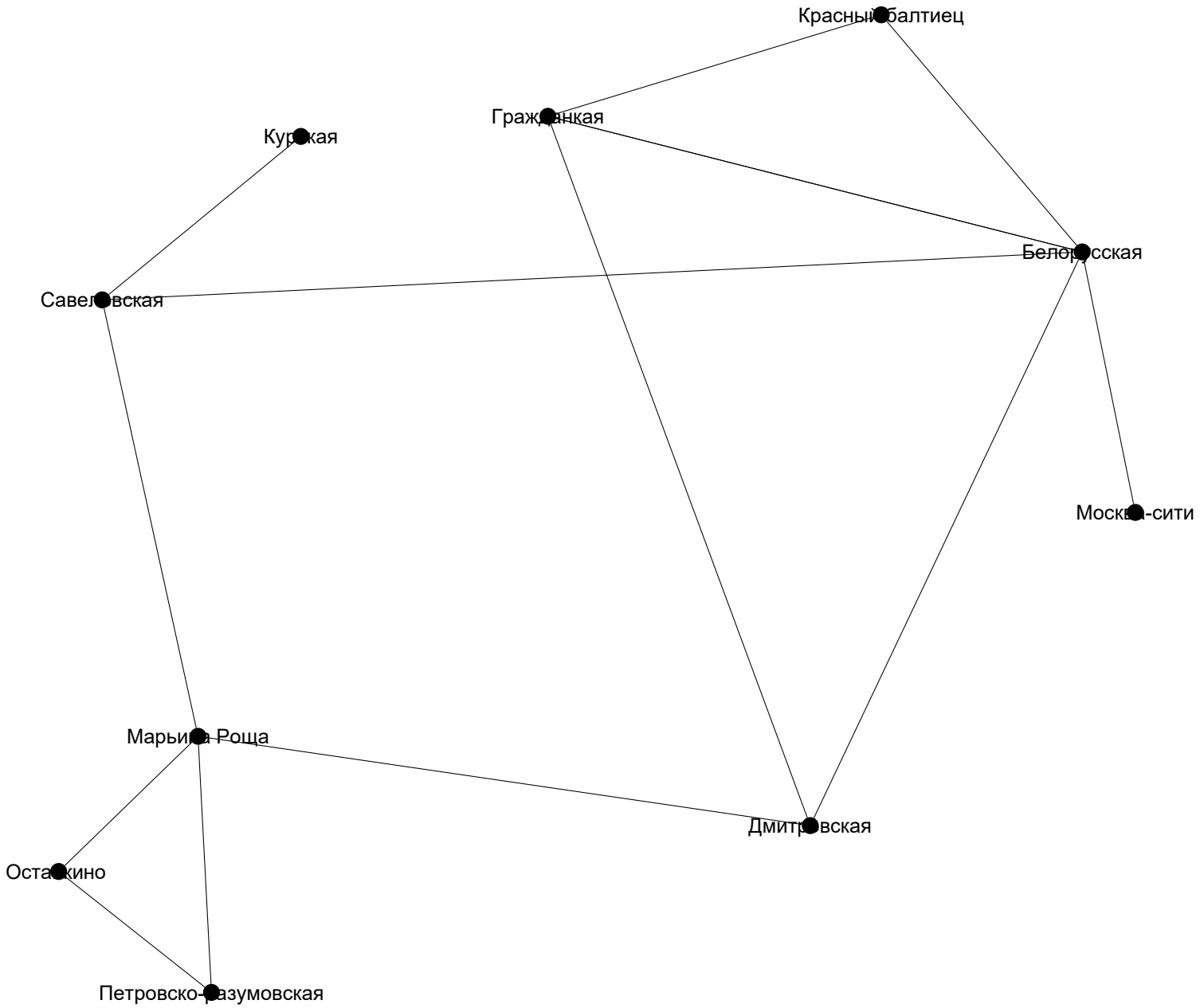
Теперь найдем 10 самых центральных станций сообщества 2. Будем использовать "суммарную мощность" для определения центральность *(то есть степень вершиннки)*:

```
community_2_nodes = df[df['Modularity class'] == largest_community]
top_10_nodes = community_2_nodes.nlargest(10, 'Суммарная мощность')

print(f"10 САМЫХ ЦЕНТРАЛЬНЫХ СТАНЦИЙ СООБЩЕСТВА {largest_community}
for i, (_, row) in enumerate(top_10_nodes.iterrows(), 1):
    print(f"{i:2}. {row['Label']:20} (ID: {row['Id']:3}, Степень: {
```

Вывод:

```
10 САМЫХ ЦЕНТРАЛЬНЫХ СТАНЦИЙ СООБЩЕСТВА 2:
 1. Марьина Роща         (ID:  48, Степень: 6)
 2. Белорусская          (ID:  90, Степень: 6)
 3. Курская              (ID:  45, Степень: 5)
 4. Гражданкая           (ID:  52, Степень: 4)
 5. Савёловская          (ID:  91, Степень: 4)
 6. Дмитровская          (ID:  50, Степень: 3)
 7. Красный Балтиец      (ID:  53, Степень: 3)
 8. Москва-Сити          (ID:  89, Степень: 3)
 9. Петровско-Разумовская (ID: 161, Степень: 3)
 10. Останкино            (ID: 162, Степень: 3)
```

Нарисуем его в gephi!


Красный Балтиец
Гражданка
Курская
Белорусская
Савеловская
Москва-сити
Марьина Роща
Дмитровская
Остаккино
Петровско-Разумовская

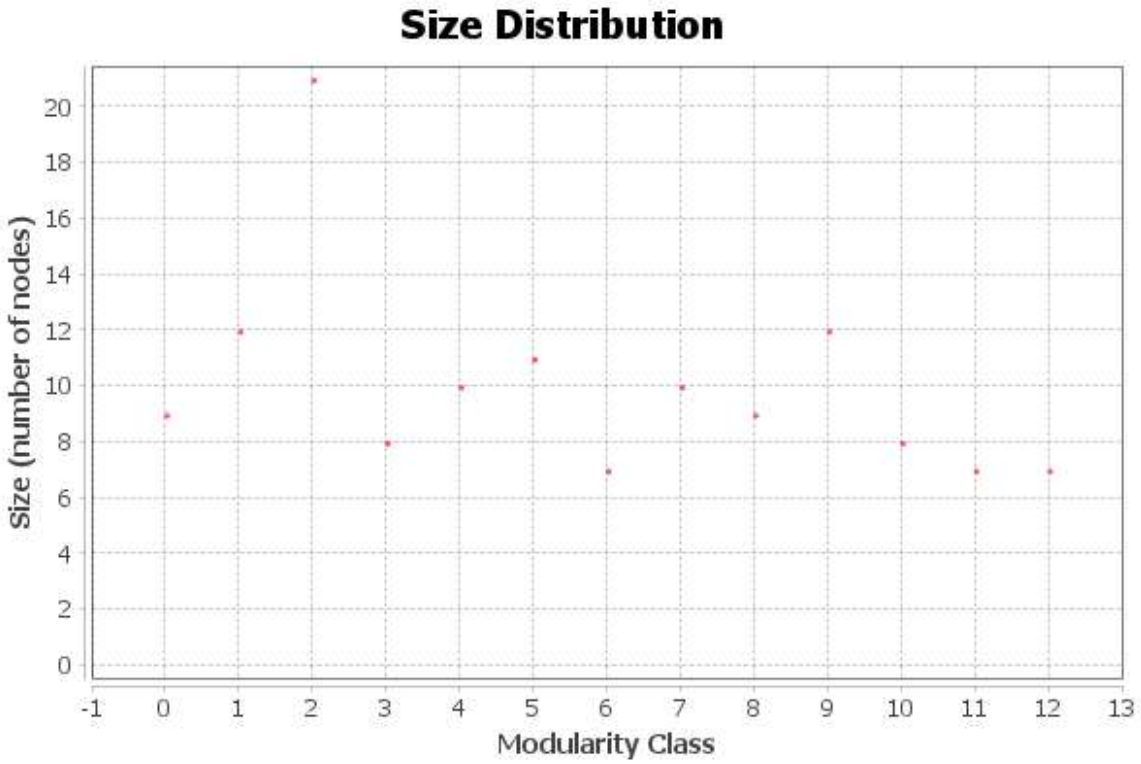
Modularity Report

Parameters:

Randomize: On
Use edge weights: On
Resolution: 1.0

Results:

Modularity: 0,822
Modularity with resolution: 0,822
Number of Communities: 13



Algorithm:

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, *Fast unfolding of communities in large networks*, in Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P1000

Resolution:

R. Lambiotte, J.-C. Delvenne, M. Barahona *Laplacian Dynamics and Multiscale Modular Structure in Networks 2009*

Окей, теперь давайте добавим веса ребрам в этом подграфе рандомно от 1 до 15 в gephi.

Мы это сделали и экспортировали файлы в формате $.csv$ с nodes и с edges из gephi! Для того чтобы их проанализировать вручную и на python. Ссылка на gephi с этим подграфом и ссылка на .csv файлики этого подграфа

```
import pandas as pd
import networkx as nx

nodes_df = pd.read_csv('nodes_task_7.csv', encoding='cp1251', delim
edges_df = pd.read_csv('edges_task_7.csv', encoding='cp1251', delim

G = nx.Graph()

for _, row in nodes_df.iterrows():
    G.add_node(row['Id'], label=row['Label'], degree=row['Суммарная

for _, row in edges_df.iterrows():
    G.add_edge(row['Source'], row['Target'], weight=row['Weight'])

start_node_id = nodes_df.loc[nodes_df['Суммарная мощность'].idxmax(

print(start_node_id)
```

Получаем вывод:

```
1
```

Значит, вершина с id-шником 1 имеет самую высокую степень. Данный id-шник соотвествует станции Белорусская. И степень у нее равна 6!

Окей, значит именно с нее нам придется запускать Дейкстру!

```
def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph.nodes()}
    distances[start] = 0
    visited = set()

    while len(visited) < len(graph.nodes()):
        current = min([node for node in graph.nodes() if node not in
                       key=lambda x: distances[x], default=None)

        if current is None:
            break

        visited.add(current)

        for neighbor in graph.neighbors(current):
            if neighbor not in visited:
                new_distance = distances[current] + graph[current][
                if new_distance < distances[neighbor]:
                    distances[neighbor] = new_distance

    return distances


shortest_distances = dijkstra(G, start_node_id)
```

Здесь мы просто реализовали алгоритм Дейкстры и запустили его по нашему графу от Белорусской.

Выведем результаты:

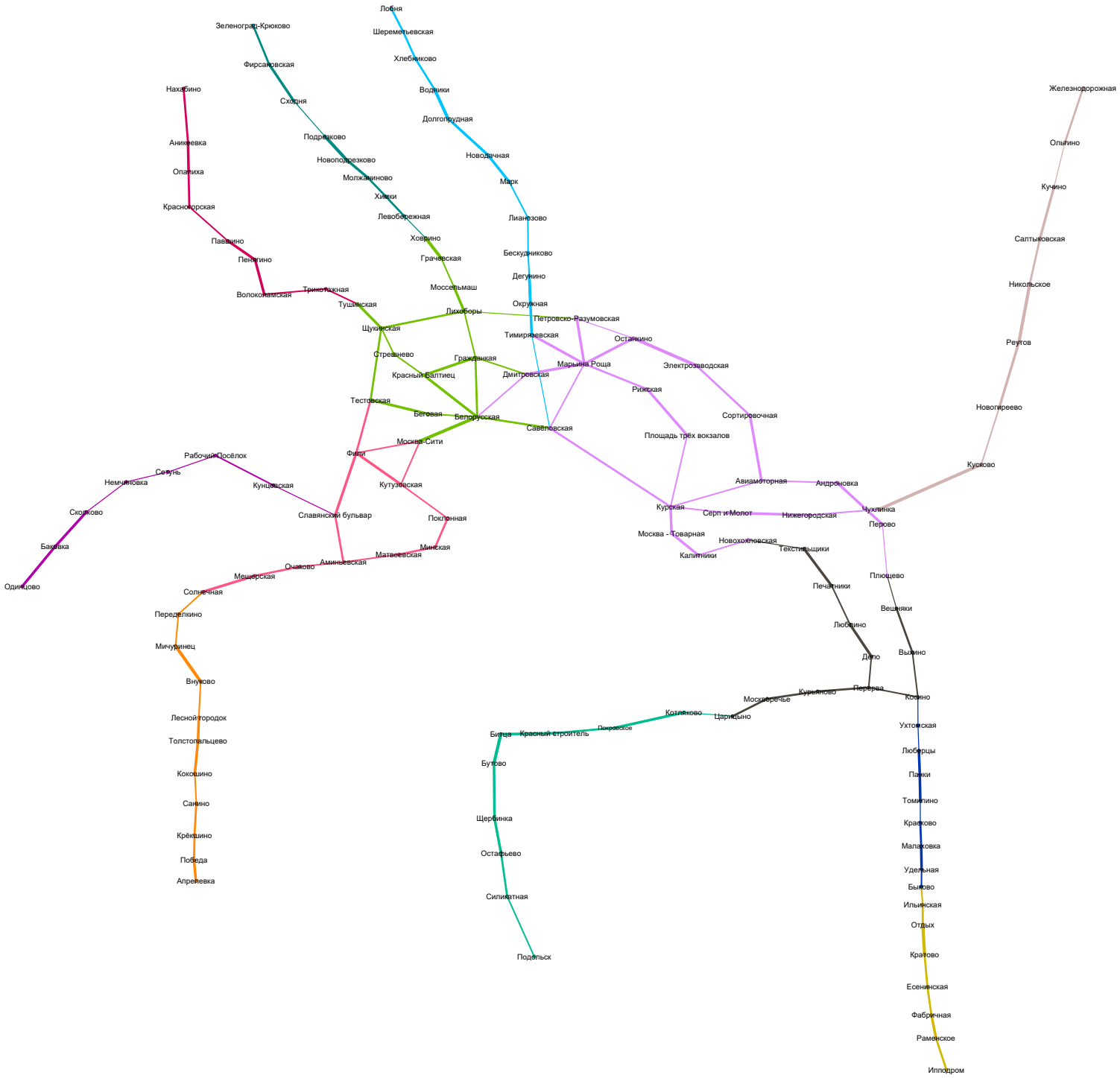
```
for node_id, distance in shortest_distances.items():
    station_name = nodes_df.loc[nodes_df['Id'] == node_id, 'Label']
    print(f"{station_name.ljust(20)} {distance if distance != float
```

Ну и получим такой список:

```
Марьина Роща         11.0
Белорусская          0
Курская              7.0
Гражданкая           9.0
Савеловская          2.0
Дмитровская          5.0
Красный балтиец      3.0
Москва-сити          7.0
Петровско-разумовская 23.0
Останкино            14.0
```

Задание решено!

Раскраска по modularity



Построим тогда мин. остовное дерево нашего подграфа алгоритмом Прима!

Имеем вершины:

- 0: Марьина Роща
- 1: Белорусская
- 2: Курская
- 3: Гражданкая
- 4: Савеловская
- 5: Дмитровская
- 6: Красный балтиец
- 7: Москва-сити
- 8: Петровско-разумовская
- 9: Останкино

Ну и имеем следующий список ребер:

- 0-9: 3.0
- 0-8: 14.0
- 0-5: 15.0
- 0-4: 9.0
- 1-4: 2.0
- 1-3: 6.0
- 1-5: 5.0
- 1-6: 3.0
- 1-7: 7.0
- 2-4: 5.0
- 3-5: 10.0
- 3-6: 8.0
- 8-9: 9.0

См. Ссылка на gephi с этим подграфом и ссылка на .csv файлики этого подграфа!

1. Выберем начальную вершину Марьина роща. Добавим ребро 0-9 и вершину 9, ибо оно мин. из доступных
2. Имеем дерево {0, 9} и доступные ребра: 0-8(14.0), 0-5(15.0), 0-4(9.0), 9-8(9.0). Выберем минимальное и добавим ребро 0-4 и вершину 4.
3. Имеем дерево {0, 9, 4}. Добавим ребро 4-1 и вершину 1! *(по той же логике, добавляем мин. вес из доступных)*
4. Имеем дерево {0, 9, 4, 1}. Добавим мин. ребро 1-6 и вершину 6.
5. Имеем дерево {0, 9, 4, 1, 6}. Добавляем мин. ребро 4-2 (вес 5.0)
6. Имеем дерево {0, 9, 4, 1, 6, 2}. Добавим мин. ребро 1-5 и вершину 5.
7. Имеем дерево {0, 9, 4, 1, 6, 2, 5}. Добавим мин. ребро 1-3 и вершину 3 (вес 6)

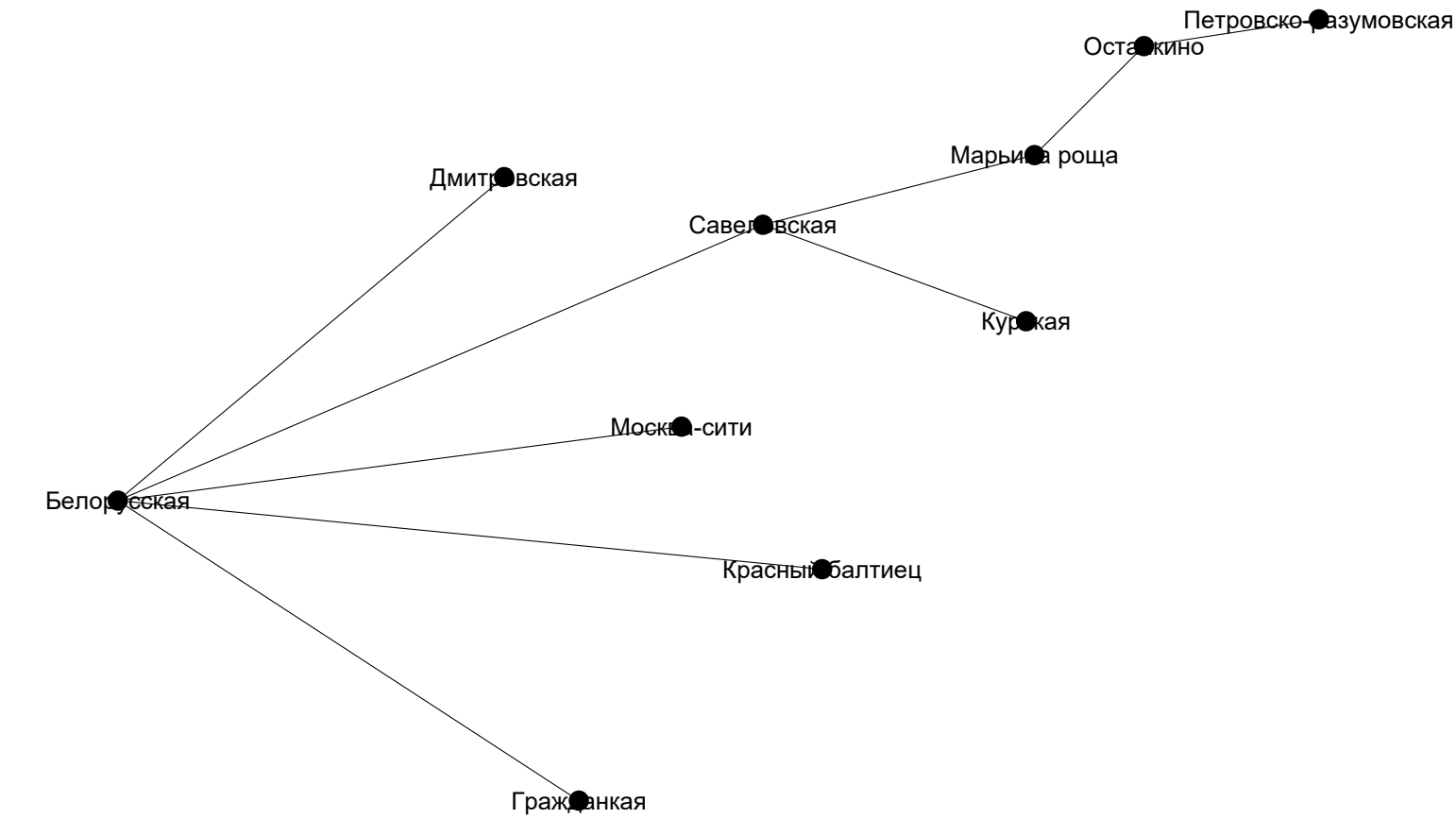
8. Имеем дерево {0, 9, 4, 1, 6, 2, 5, 3}. Добавим мин. ребро 1-7 и вершину 7 (вес 7)
9. Имеем дерево {0, 9, 4, 1, 6, 2, 5, 3, 7}. Добавим мин. ребро 9-8 и вершину 8 (вес 9)

Итог: Мы построили мин. остовное дерево и его вес суммарный 49. *(просто складывал после каждого шага)*

Ребра дерева:

1. Марьина Роща - Останкино (вес 3.0)
2. Марьина Роща - Савеловская (вес 9.0)
3. Савеловская - Белорусская (вес 2.0)
4. Белорусская - Красный балтиец (вес 3.0)
5. Савеловская - Курская (вес 5.0)
6. Белорусская - Дмитровская (вес 5.0)
7. Белорусская - Гражданкая (вес 6.0)
8. Белорусская - Москва-сити (вес 7.0)
9. Останкино - Петровско-разумовская (вес 9.0)

Построим график в gephi:


Петровско-Разумовская
Останкино
Марьина роща
Дмитровская
Савеловская
Курская
Москва-сити
Белорусская
Красный Балтиец
Гражданка